Docket No.: SI-0052 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:
	: Customer No.: 34610
Sang Ik JUNG and Seok Jin YOON	:
	:
Serial No.: New U.S. Patent Application	:
	:
Filed: December 10, 2003	:
	:
For: CACHE FLUSH SYSTEM AND METHOD THEREOF	

TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

U.S. Patent and Trademark Office
2011 South Clark Place
Customer Window
Crystal Plaza Two, Lobby, Room 1B03
Arlington, Virginia 22202

Sir:

At the time the above application was filed, priority was claimed based on the following application:

Korean Patent Application No. 10-2002-0083582, filed on December 24, 2002.

A copy of each priority application listed above is enclosed.

Respectfully submitted,
FLESHNER & KIM, LLP

Carl R. Wesolowski
Registration No. 40,372

P.O. Box 221200
Chantilly, Virginia 20153-1200
703 502-9440 CRW/par
Date: December 10, 2003

Please direct all correspondence to Customer Number 34610

대한민국특허청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2002-0083582
Application Number

출 원 년 월 일 : 2002년 12월 24일
Date of Application DEC 24, 2002

출 원 인 : 엘지전자 주식회사
Applicant(s) LG Electronics Inc.

2003 년 10 월 09 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0004
【제출일자】	2002.12.24
【발명의 명칭】	캐시 플러시 시스템 및 방법
【발명의 영문명칭】	Cache Flush System And Method
【출원인】	
【명칭】	엘지전자 주식회사
【출원인코드】	1-2002-012840-3
【대리인】	
【성명】	김영철
【대리인코드】	9-1998-000040-3
【포괄위임등록번호】	2002-027003-6
【대리인】	
【성명】	김순영
【대리인코드】	9-1998-000131-1
【포괄위임등록번호】	2002-027004-3
【발명자】	
【성명의 국문표기】	정상익
【성명의 영문표기】	JUNG,Sang Ik
【주민등록번호】	691106-1654211
【우편번호】	440-320
【주소】	경기도 수원시 장안구 율전동 현대A 305-105
【국적】	KR
【발명자】	
【성명의 국문표기】	윤석진
【성명의 영문표기】	YOON,Seok Jin
【주민등록번호】	731025-1029430
【우편번호】	463-010
【주소】	경기도 성남시 분당구 정자동 상록마을 상록우성 아파트 309동 801호
【국적】	KR

【취지】 특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 대리인
김영철 (인) 대리인
김순영 (인)

【수수료】

【기본출원료】	20	면	29,000 원
【가산출원료】	27	면	27,000 원
【우선권주장료】	0	건	0 원
【심사청구료】	0	항	0 원
【합계】	56,000 원		

【첨부서류】 1. 요약서·명세서(도면)_1통

【요약서】

【요약】

본 발명은 캐시 플러시 시스템 및 방법에 관한 것으로, 특히 멀티 프로세서 시스템에서 특정 이벤트에 대해 캐시 메모리에서 고속 자동화된 캐시 플러시가 이루어지도록 한 캐시 플러시 시스템 및 방법에 관한 것이다.

본 발명은 최대 각 프로세서의 2차 캐시 메모리 크기만큼의 메모리 리드만을 수행함으로써, 프로세서 버스의 부하를 최소화할 수 있고, 캐시 플러시 자체가 특정 이벤트에 의하여 직접 트리거되는 캐시 플러시 알고리즘에 의하여 수행됨으로써, 캐시 플러시의 특정 이벤트에 대한 동시성을 보장할 수 있어 고속 자동화된 캐시 플러시가 이루어질 수 있게 하는 효과가 있다.

【대표도】

도 5

【명세서】

【발명의 명칭】

캐시 플러시 시스템 및 방법{Cache Flush System And Method}

【도면의 간단한 설명】

도 1은 종래의 멀티 프로세서 시스템을 나타낸 도면.

도 2는 1차 데이터 캐시 메모리 구조를 나타낸 도면.

도 3은 MESI 프로토콜을 나타낸 도면.

도 4는 본 발명이 적용되는 멀티 프로세서 시스템을 나타낸 도면.

도 5는 도 4에 있어 본 발명의 실시예에 따른 캐시 플러시 시스템을 나타낸 도면.

도 6은 도 5에 있어 밸리드 어레이부를 나타낸 도면.

도 7은 도 6에 있어 디캔드를 나타낸 도면.

도 8은 도 6에 있어 디코아를 나타낸 도면.

도 9는 도 5에 있어 태그 저장부를 나타낸 도면.

도 10은 도 9에 있어 어드레스 맵핑을 개념적으로 나타낸 도면.

도 11은 본 발명의 실시예에 따른 캐시 플러시 방법을 나타낸 순서도.

도 12는 도 11에 있어 갱신 과정을 나타낸 순서도.

도 13은 도 7의 디캔드를 이용한 정치 동작을 개념적으로 나타낸 도면.

도 14는 도 11에 있어 캐시 플러시 과정을 나타낸 순서도.

도 15는 도 8의 디코아를 이용한 어드레스 배열 동작을 개념적으로 나타낸 도면.

* 도면의 주요 부분에 대한 부호의 설명 *

6 : 캐시 메모리 7 : 프로세서

8 : 메모리 제어부 9 : 메인 메모리

10 : 버스 스누프부 11 : 타 시스템 자원

12 : 프로세서 버스 20 : 태그 저장부

21 : 태그 램부 22 : 매치 로직부

30 : 밸리드 어레이부 31 : 디캔드

32 : 디코아 40 : 캐시 플러시부

41 : 캐시 플러시 수행기 42 : 이벤트 검출기

43 : 캐시 버스 마스터

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<26> 본 발명은 캐시 플러시(Cache Flush) 시스템 및 방법에 관한 것으로, 특히 멀티 프로세서 시스템(Multi-Processor System)에서 특정 이벤트(Event)에 대해 캐시 메모리에서 고속 자동화된 캐시 플러시가 이루어지도록 한 캐시 플러시 시스템 및 방법에 관한 것이다.

<27> 일반적으로, 프로세서 시스템(Processor System)에서는 프로세서에 의한 메인 메모리로의 억세스를 고속화하기 위해서, 프로세서가 필요로 하는 데이터를 일시적으로 저장하는 캐시 메모리(Chache)가 사용되고 있다. 통상, 캐시 메모리는 캐시 태그(Chache Tag)라고 하는 관리 정보를 유지하여, 캐시 메모리에 저장된 캐시 블록(Chache Block)의 데이터가 메인 메모리 중의 어떤 데이터인지 및 프로세서에 의해서 변경되어 메인 메모리의 내용과 다른 상태(변경 상태(Modified State) 또는 더티 상태(Dirty State))에 있는지 등을 관리하고 있다.

<28> 한편, 프로세서를 복수 설치한 시스템, 즉 멀티 프로세서 시스템에서는 캐시 메모리도 복수이지만, 해당 캐시 메모리 사이에서 메모리 일관성(Memory Coherency) 또는 데이터의 일관성(Data Coherency)을 보증하기 위해, 해당 각 캐시 메모리는 스누프 기능(Snoop Mechanism)을 갖는다. 해당 스누프 기능은 프로세서 버스 명령이 자신의 캐시 메모리에 저장된 데이터에 영향을 미치는지의 여부 또는 자신의 캐시 메모리에 저장된 데이터를 응답으로 돌려 주어야 하는지의 여부 등을 검사하고, 필요하면, 대응하는 데이터의 무효화(Invalide) 등을 행하는 것이다.

<29> 캐시 메모리에는 카피 백형(Copy Back)형, 라이트 스루(Wirte Through)형이 있지만, 프로세서에 의한 데이터 갱신을 즉시 메인 메모리에 반영시키지 않고서 어느 정도의 기간 내부에 유지하는 것을 계속하는 카피 백형 캐시 메모리의 경우, 프로세서에 의해 캐시 메모리 상에서 변경되어 메인 메모리의 내용과 다른 데이터를 적극적으로 메인 메모리에 재기록(Write Back) 조작을 필요로 한다. 예컨대, 스누프 기능을 가지지 않은 입출력 장치(I/O Device)에 대하여 캐시 메모리가 저장하는 데이터를 전송하는 경우 등에 재기록이 필요하다, 캐시 메모리가 저장하는 데이터 중 변경되어 있는 데이터를 메인 메모리에 재기록 조작을 캐시 플러시(Cache Flush)라고 한다. 그리고, 더티 상태에 있는 캐시 블록을 더티 블록이라고 한다.

<30> 이 캐시 플러시는 스누프 기능을 가지지 않은 입출력 장치에 대한 데이터의 전송시 이외에도, 고장 감내형(fault tolerant) 또는 복제형(replicant) 시스템에서 유용하다. 즉, 시스템에 어떠한 장애가 발생했을 때, 미리 획득된 체크포인트로부터 처리를 재개하는 체크포인트 방식 시스템의 경우, 체크포인트의 시점에서, 캐시 메모리 중에만 존재하는 변경된 데이터를 메인 메모리에 재기록할 필요가 있다.

<31> 이러한 캐시 플러시는 통상, 캐시 플러시 명령(Instruction)을 포함하는 소프트웨어에 기초하여 실행된다. 소프트웨어를 실행함으로써, 프로세서는 캐시 태그의 내용을 참조하여, 캐시 블록마다 더티 블록인지의 여부를 판단하며, 더티이면 대응하는 캐시 블록의 데이터를 메인 메모리에 재기록하는 캐시 플러시 동작을 행한다.

<32> 이와 같이, 프로세서는 부여된 프로그램에 따라서, 하나 하나 모든 캐시 블록에 대해서, 상태를 판정하여 더티인 경우에는 메인 메모리로의 재기록, 즉 캐시 플러시 동작을 실행하지 않으면 안된다. 모든 캐시 블록에 대해서 처리를 행하기 위해서는 매우 많은 시간을 필요로 한다.

<33> 이하, 도 1을 참조하여 종래의 멀티 프로세서 시스템을 설명한다.

<34> 도 1은 종래의 멀티 프로세서 시스템을 나타낸 도면이다.

<35> 종래의 멀티 프로세서 시스템은 프로세서 버스(Processor Bus)에 연결된 다수의 프로세서(CPU : Central Processing Unit)(5)와, 해당 프로세서(5) 각각에 연결된 캐시 메모리(1)와, 상기 프로세서 버스에 연결된 메모리 제어부(2)와, 해당 메모리 제어부(2)의 제어를 받은 메인 메모리(3) 및 상기 프로세서 버스에 연결된 타 시스템 자원(Other System Resources)(4)으로 구성된다.

<36> 여기서, 상기 각 프로세서(5)는 내부, 외부의 백 사이드(Back Side) 또는 두 곳 모두에 소정의 캐시 메모리(1)를 구비하고 있다. 이때, 내부의 캐시 메모리를 1 차 캐시 메모리(Level 1 Cache)라 하며, 외부의 캐시 메모리를 2차 캐시 메모리(Level 2 Cache)라 한다.

<37> 그리고, 상기 각 프로세서(5)는 공통의 프로세서 버스를 통하여 상호 연결되어 있으며, 해당 프로세서 버스를 통하여 명령어 인출(Instruction Fetch) 및 데이터 로드/저장(Load/Store)을 위해 상기 메인 메모리(3)에 접근할 수 있다. 이때, 상기 메인 메모리(3)의 접근은 통상 메모리 제어부(2)를 통하여 이루어진다.

<38> 한편, 상기 종래의 멀티 프로세서 시스템은 상술한 기본 자원 이외에 자신의 특정한 기능을 수행하기 위하여 입출력 장치 등의 타 시스템 자원(4)을 통해 연결되는데, 32 비트 어드레스(Address)와 64 비트 데이터 버스를 구비한다고 할 때, 해당 버스에 정합되는 모든 디바이스, 즉 상기 프로세서(5), 메모리 제어부(2) 및 타 시스템 자원(4)은 이와 동일한 정합을 가져야 하며, 상기 각 프로세서(5)의 캐시 메모리(1)도 이를 근거로 하여 그 구성을 갖는다.

<39> 그리고, 상기 각 프로세서(5)는 내부에 32 키로바이트(KB)의 1차 명령어 캐시 메모리(L1 Instruction Cache)와 32 KB의 1차 데이터 캐시 메모리(L1 Data Cache)를 갖고, 외부의 백 사이드에 1 메가바이트(MB)의 2차 캐시 메모리를 갖는다.

<40> 이하, 도 2를 참조하여 상기 1차 데이터 캐시 메모리 구조를 설명한다.

<41> 도 2는 1차 데이터 캐시 메모리 구조를 나타낸 도면이다.

<42> 상기 1차 데이터 캐시 메모리는 태그 램(Tag RAM : Random Access Memory)과 데이터 램(Data RAM)으로 구성되는데, 상기 1차 데이터 캐시 메모리는 8-웨이 세트-어쏘시에이티브 맵핑(8-Way Set-Associative Mapping)을 채택한 것이며, 8개의 각 캐시 블록은 4 개의 워드

(Word)(W0~W3, 각 64 비트)와 그에 상응하는 어드레스 태그(20 비트)로 구성된다. 또한, 각 캐시 블록의 상태를 표시하기 위한 3 개의 상태 정보 비트, 즉 밸리드 상태 비트(V : Valid), 변경 상태 비트(M : Modified) 및 공유 상태 비트(S : Shared)를 갖는다.

<43> 그리고, 상기 1차 명령어 캐시 메모리는 상기 1차 데이터 캐시 메모리와 마찬가지로 구성되나, 각 캐시 블록의 상태 정보 비트로 밸리드 상태 비트만을 가진다. 또한, 상기 2차 캐시 메모리는 명령어 또는 데이터에 상관없이 데이터 램에 저장하며, 다이렉트 맵핑(Direct Mapping)을 채택하므로, 32 키로바이트의 세트(Set)로 구성되면 각 세트의 크기는 1 블록이다.

<44> 종래에는 일반적으로 상기 1차 캐시 메모리와 2차 캐시 메모리는 캐시의 효율을 높이기 위하여 라이트 원칙(Write Policy)으로서 라이트 백(Write-back) 방식을 채택하지만, 이로 인하여 프로세서(5) 상호 간 및 프로세서(5)와 입출력 디바이스간 메모리 일관성(Memory Coherency) 문제가 대두된다. 이를 관리하기 위하여 각 프로세서(5)의 캐시 제어부(Cache Controller)는 MESI(Modified/Exclusive/Shared/Invalid) 프로토콜을 사용하는데, 도 3은 MESI 프로토콜을 나타낸 도면이다.

<45> 도 3에 도시된 바와 같이, 상기 MESI 프로토콜의 상태는 변경(Modified) 상태, 배타(Exclusive) 상태, 공유(Shared) 상태 및 인밸리드(Invalid) 상태 등이 있으며, 각 캐시 블록의 상태 정보 비트를 조합하여 나타낼 수 있다. 즉, 상기 인밸리드 상태는 상태 정보 비트에서 밸리드 상태 비트(V) = 0, 변경 상태 비트(M) = 0, 공유 상태 비트(S) = 0인 경우이고, 상기 변경 상태는 상태 정보 비트에서 밸리드 상태 비트(V) = 1, 변경 상태 비트(M) = 1, 공유 상태 비트(S) = 0인 경우이고, 상기 공유 상태는 상태 정보 비트에서 밸리드 상태 비트(V) = 1, 변경 상태 비트(M) = 0, 공유 상태 비트(S) = 1인 경우이며, 상기 배타 상태는 상태 정보 비트에서 밸리드 상태 비트(V) = 1, 변경 상태 비트(M) = 0, 공유 상태 비트(S) = 0인 경우이다.

<46> 상기 MESI 프로토콜에 의하여 상기 각 프로세서(5)에서 별도로 관리되고 있던 캐시 메모리(1)는 멀티 프로세서 시스템의 특정 이벤트(Event)에 대하여 변경 상태 또는 더티 상태에 있는 캐시 블록을 상기 메인 메모리(3)에 카피백, 즉 재기록하는 캐시 플러시 동작을 함으로써, 메모리 일관성을 보전한다. 이때의 절차를 살펴본다. 특정 이벤트가 발생하면, 상기 각 프로세서(5)는 해당 이벤트에 대한 예외 루틴(Exception Routine)을 수행한다. 이 루틴의 수행 도중 적당한 시간을 택하여 캐시 플러시 동작을 수행하게 되는데, 이에 적용되는 방법은 통상 2차 캐시 메모리 크기의 2 배에 상당하는 연속된 메모리 영역을 로드함으로써 1차 데이터 캐시 메모리 및 2차 캐시 메모리의 변경 캐시 블록이 캐시 플러시되게 하는 것이다.

<47> 그런데, 대개의 경우 캐시 플러시를 요구하는 이벤트는 긴급성을 나타내며, 그 처리 또한 신속성을 요구하지만, 종래의 캐시 플러시 방법에서는 프로세서 버스에 연결되어 있는 모든 프로세서가 동시에 최소한 2차 캐시 메모리 크기만큼의 메모리 리드(Read)를 수행하기 때문에 프로세서 버스의 부하가 불필요하게 증가되고, 캐시 플러시가 각 프로세서의 예외 루틴에 의해 수행되기 때문에 실제로 캐시 플러시가 일어나는 시점은 특정 이벤트가 발생하고 상당 시간이 흐른 뒤이므로, 신속하게 캐시 플러시 동작을 수행할 수 없는 문제점이 있었다.

【발명이 이루고자 하는 기술적 과제】

<48> 상술한 바와 같은 문제점을 해결하기 위한 것으로, 본 발명의 목적은 최대 각 프로세서의 2차 캐시 메모리 크기만큼의 메모리 리드만을 수행함으로써, 프로세서 버스의 부하를 최소화하도록 하는데 있다.

<49> 그리고, 본 발명의 다른 목적은 캐시 플러시 자체가 특정 이벤트에 의하여 직접 트리거(Trigger)되는 캐시 플러시 알고리즘(Cache Flush Algorithm)에 의하여 수행됨으로써, 캐시 플러시의 특정 이벤트에 대한 동시성을 보장하도록 하는데 있다.

【발명의 구성 및 작용】

<50> 상술한 바와 같은 목적을 해결하기 위하여, 본 발명의 캐시 플러시 시스템은 각 프로세서의 캐시 메모리에서 배타 또는 변경의 밸리드 상태 캐시 블록에 대하여 갱신 동작 및 캐시 플러시 동작을 위한 캐시 블록 정보 및 인덱스 정보를 제공하는 밸리드 어레이부와; 태그 램을 저장하고 갱신 동작 및 캐시 플러시 동작을 위한 매치 어드레스 정보 및 태그 정보를 제공하는 태그 저장부와; 프로세서 버스를 감시하고 상기 각 캐시 메모리의 상태를 추적하여 상기 태그 저장부와 밸리드 어레이부에 대한 갱신 동작을 수행하는 버스 스누프부와; 시스템의 특정 이벤트를 감지하고 상기 캐시 메모리의 밸리드 상태 캐시 블록에 대해 캐시 플러시를 수행하게 하는 리드 트랜잭션을 생성하여 출력하는 캐시 플러시부를 포함하여 이루어진 것을 특징으로 한다.

<51> 또한, 본 발명의 캐시 플러시 방법은 프로세서 버스의 트랜잭션을 감시하고 각 프로세서 캐시 메모리의 상태를 추적하여 태그 램과 밸리드 어레이에 대한 갱신 동작을 수행하는 과정과; 특정 이벤트를 감지하고 상기 태그 램과 밸리드 어레이를 이용해 리드 트랜잭션을 생성 출력하여 상기 캐시 메모리 중 밸리드 상태의 캐시 블록에 대한 캐시 플러시 동작을 수행하는 과정을 포함하여 이루어진 것을 특징으로 한다.

<52> 이하, 본 발명의 실시예를 첨부한 도면을 참조하여 상세하게 설명하면 다음과 같다.

<53> 먼저 도 4를 참조하여 본 발명이 적용되는 멀티 프로세서 시스템을 설명한다.

<54> 도 4는 본 발명이 적용되는 멀티 프로세서 시스템을 나타낸 도면이다.

<55> 본 발명이 적용되는 멀티 프로세서 시스템은 버스 스누프 기능(Bus Snoop)을 갖는 카피백형의 캐시 메모리(6)를 구비하여 프로세서 버스(12)에 연결된 다수의 프로세서(CPU)(7)와, 상기 프로세서 버스(12)에 연결된 메모리 제어부(8)와, 해당 메모리 제어부(8)의 제어를 받는 메인 메모리(9)와 상기 프로세서 버스(12)에 연결된 타 시스템 자원(11) 및 상기 프로세서 버스(12)에 연결된 캐시 플러시 시스템(100)을 포함하여 이루어진다. 특히, 도 4에서는 상기 프로세서(7)가 4개(프로세서0~프로세서3)이고 이에 따라 캐시 메모리(6)가 4개(캐시 메모리0~캐시 메모리3)인 경우를 예시한다.

<56> 이하, 도 5를 참조하여 상기 캐시 플러시 시스템(100)을 설명한다.

<57> 도 5는 도 4에 있어 캐시 플러시 시스템을 나타낸 도면이다.

<58> 상기 캐시 플러시 시스템(100)은 특정 이벤트에 대하여 캐시 플러시 동작을 수행하고 각 캐시 메모리(6)의 상태를 추적 관리하는 것으로, 밸리드 어레이부(Valid Array)(30), 버스 스누프부(Bus Snooper)(10), 태그 저장부(20) 및 캐시 플러시부(40)를 포함하여 이루어진다.

<59> 먼저, 도 6을 참조하여 상기 밸리드 어레이부(30)를 설명한다.

<60> 도 6은 도 5에 있어 밸리드 어레이부를 나타낸 도면이다.

<61> 상기 밸리드 어레이부(30)는 디캔드(DCAND : Divide & Conquer AND tree)(31)와 디코어(DCOR : Divide & Conquer OR tree)(32)를 구비하여 상기 각 캐시 메모리(6)에서 배타 상태 또는 변경 상태, 즉 밸리드 상태의 캐시 블록에 대한 갱신 동작(Update Algorithm) 및 캐시 플

러시 동작(Cache Flush Algorithm)을 위한 캐시 블록 정보를 상기 버스 스누프부(10)로 제공하고 인덱스(Index) 정보를 상기 캐시 플러시부(40)로 제공한다.

<62> 구체적으로 설명하면, 상기 캐시 플러시 시스템(100)은 시스템에 이벤트가 발생했을 때, 상기 각 캐시 메모리(6)에서 배타 상태 또는 변경 상태의 캐시 블록에 대해 캐시 플러시 동작을 수행하는데, 이를 효율적으로 관리하기 위하여 밸리드 어레이부(30)를 구비한다.

<63> 상기 밸리드 어레이부(30)는 밸리드 어레이를 갖는데, 해당 밸리드 어레이는 각 요소가 해당 캐시 블록이 배타 상태 또는 변경 상태, 즉 밸리드 상태에 있는지를 나타내는 밸리드 비트(Valid bit)로 이루어진 3차원 어레이로서, 그 한 축은 상기 프로세서(7)이다. 따라서 각 프로세서(7)에 대하여 도 6에 도시된 바와 같은 2차원 밸리드 어레이를 갖는다. 여기서 밸리드 비트가 '1'이면 밸리드 상태로서 클리어(Clear)해야하고 이에 따라 캐시 플러시 동작이 수행된다. 반면에 밸리드 비트가 '0'이면 밸리드 상태가 아닌, 즉 인밸리드 상태로서 클리어할 필요가 없고 이에 따라 캐시 플러시 동작이 수행되지 않는다.

<64> 그리고, 상기 밸리드 어레이부(30)는 갱신 동작 및 캐시 플러시 동작이 효과적으로 수행될 수 있도록 로직 트리(Logic Tree), 즉 상기 디캔드(31)와 디코아(32)를 구현한다. 특정 세트의 디캔드(31)와 특정 캐시 블록의 간략화된 디코아(32)가 도 7과 도 8에 도시되어 있다. 여기서, 도 7은 도 6에 있어 디캔드를 나타낸 도면이고, 도 8은 도 6에 있어 디코아를 나타낸 도면이다. 도 7에 도시된 디캔드(31)는 한 캐시 세트 인덱스에 대한 모든 밸리드 비트에 관한 것으로 캐시 블록 정보를 상기 버스 스누프부(10)로 제공하고, 도 8에 도시된 디코아(32)는 모든 캐시 세트에 있는 한 캐시 블록에 대한 15 비트 중에서 예시로서 5 비트 인덱스를 갖는 밸리드 비트에 관한 것으로 인덱스 정보를 상기 캐시 플러시 수행기(41)로 제공한다.

<65> 이하, 도 9와 도 10을 참조하여 상기 태그 저장부(20)를 설명한다.

<66> 도 9는 도 5에 있어 태그 저장부를 나타낸 도면이고, 도 10은 도 9에 있어 어드레스 맵핑을 개념적으로 나타낸 도면이다.

<67> 상기 태그 저장부(20)는 태그 램부(Tag RAM)(21)와 매치 로직부(Match Logic)(22)를 구비하여 태그 램을 저장하고 갱신 동작 및 캐시 플러시 동작을 위한 매치 어드레스 정보를 상기 버스 스누프부(10)로 제공하고 태그 정보를 상기 캐시 플러시부(40)로 제공하는데, 상기 태그 램부(21)는 특정 인덱스에 대하여 일정한 수만큼의 태그를 저장하고, 상기 매치 로직부(22)는 상기 태그 램부(21)와 정합하여 상기 매치 어드레스 정보를 출력해 프로세서 버스 어드레스와 해당 어드레스 인덱스의 태그간 매치 여부를 상기 버스 스누프부(10)로 제공하고, 상기 태그 정보를 출력하여 리드 트랜잭션(Read Transaction)의 어드레스를 구성할 태그를 상기 캐시 플러시부(10)로 제공한다.

<68> 구체적으로 설명하면, 상기 태그 저장부(20)에 구비된 태그 램부(21)은 특정 인덱스에 대하여 일정한 수 만큼의 태그를 저장하는데, 이때 해당 인덱스의 크기와 태그의 크기는 도 10에 도시된 바와 같이 역비례 관례에 있다. 즉, 32 KB 8-웨이 세트-어쏘시에이티브 1차 데이터 캐시 메모리와 1 MB 다이렉트 2차 캐시 메모리의 구성인 경우, 인덱스는 15 비트이며 태그는 12 비트이다.

<69> 도 9는 상기 태그 저장부(20)의 전체적인 구성을 나타내는데, 상기 태그 램부(21)는 상기 각 프로세서(7)에 대해 8개의 태그 램(태그 램0~태그 램7)을 구비한다. 그리고, 상기 매치 로직부(22)는 상기 태그 램부(21)과 직접 연결되어 해당 각 태그 램과 정합하는 매치 로직(매치 로직0~매치 로직7)을 구비하고 해당 태그 램의 내용을 갱신하거나 캐시 플러시를 위한 어드레스를 추출하도록 한다. 또한, 상기 매치 로직부(22)는 현재 프로세서 버스(12)에서 트랜잭션(Transaction)이 진행되고 있는 어드레스에 대하여 각 캐시 메모리(6)에 매치하는 어

드레스가 존재하는지의 여부를 판단해 상기 버스 스누프부(10)에 알려 태그 램 갱신 동작이 정상적으로 수행될 수 있게 한다.

<70> 즉, 도 9에서 트랜잭션 어드레스 중 어드레스A(12:28)는 상기 태그 램부(21)의 태그 램으로 입력되고 어드레스A(0:11)는 상기 매치 로직부(22)의 매치 로직으로 입력되는데, 예컨대 매치 로직0은 태그0과의 매치여부를 매치(0,0)로 출력하고, 매치 로직1은 태그1과의 매치여부를 매치(0,1)로 출력하고, …, 매치 로직7은 태그7과의 매치여부를 매치(0,7)로 출력하며, 각각의 매치 출력을 OR하여 그 값이 '1'이면 해당 어드레스가 매치됨을 상기 버스 스누프부(10)로 알린다. 그리고, 상기 태그 램부(21)는 상기 캐시 플러시부(40)로부터 인덱스를 제공받고 매치 로직부(22)는 해당 인덱스와 대응하는 태그 정보를 상기 캐시 플러시부(40)로 제공한다.

<71> 이하, 상기 버스 스누프부(10)를 설명하면, 상기 버스 스누프부(10)는 상기 프로세서 버스(12)에 연결되어 해당 버스를 감시하고 자체의 갱신 동작에 의하여 상기 각 캐시 메모리(6)의 상태를 추적해 상기 태그 램부(21)과 밸리드 어레이부(22)에 대한 갱신 동작을 수행한다. 이때, 상기 버스 스누프부(10)는 상기 태그 램부(21)에 대한 갱신 동작 수행시에 상기 디캔드(31)의 캐시 블록 정보를 이용한 정치 동작을 구동하여 정확한 갱신이 일어날 수 있도록 한다.

<72> 마지막으로, 상기 캐시 플러시부(40)를 설명하면, 상기 캐시 플러시부(40)는 시스템의 특정 이벤트를 감지하고 자체의 캐시 플러시 동작에 의하여 각 프로세서(7)로 하여금 밸리드 상태의 캐시 블록에 대해 캐시 플러시하게 하는 리드 트랜잭션을 생성하여 출력한다.

<73> 구체적으로 설명하면, 상기 캐시 플러시부(40)는 이벤트 검출기(42), 캐시 플러시 수행기(41) 및 캐시 버스 마스터(43)를 포함하여 이루어지는데, 상기 이벤트 검출기(42)는 시스템에 대한 특정 이벤트의 발생 여부를 검출하고, 상기 캐시 플러시 수행기(41)는 상기 캐시 플러

시 동작을 수행하여 상기 인덱스 정보와 상기 태그 정보로 매핑된 어드레스를 가지고 리드 트랜잭션을 생성하고, 상기 캐시 버스 마스터(43)는 상기 생성된 리드 트랜잭션을 상기 프로세서 버스(12)로 출력하여 상기 각 프로세서(7)로 전달해 상기 캐시 메모리(6)에 대한 캐시 플러시가 이루어지도록 한다.

<74> 이때, 상기 캐시 플러시 수행기(41)는 상기 디코아(32)의 인덱스 정보를 이용한 어드레스 배열 동작(A^3 : Address Arrangement Algorithm)을 구동하여 상기 인덱스를 추출하고 해당 인덱스를 상기 태그 램부(21)에 제공해 상기 매치 로직부(22)로부터 태그 정보를 획득하고 해당 인덱스와 태그를 병합하여 상기 어드레스를 매핑한다.

<75> 이하, 도 11을 참조하여 본 발명의 실시예에 따른 캐시 플러시 방법을 설명한다.

<76> 도 11은 본 발명의 실시예에 따른 캐시 플러시 방법을 나타낸 순서도이다.

<77> 먼저, 버스 스누프부(10)는 프로세서 버스(12)의 트랜잭션을 감시하고 자체 갱신 동작에 의해 각 프로세서(7) 캐시 메모리(6)의 상태를 추적하여 태그 램부(21)의 태그 램과 밸리드 어레이부(20)의 밸리드 어레이에 대한 갱신 동작을 수행한다(S10).

<78> 그리고, 캐시 플러시부(40)는 특정 이벤트를 감지하고 상기 태그 램과 밸리드 어레이를 이용해 리드 트랜잭션을 생성 출력하여 상기 캐시 메모리(6) 중 밸리드 상태의 캐시 블록에 대한 캐시 플러시 동작을 수행한다(S20).

<79> 이하, 도 12를 참조하여 상기 갱신 과정(S10)을 설명한다.

<80> 도 12는 도 11에 있어 갱신 과정을 나타낸 순서도이다.

<81> 먼저, 상기 버스 스누프부(10)는 상기 프로세서 버스(12) 상에서 새로운 트랜잭션이 시작하는지를 실시간 감시한다(S101).

<82> 상기 감시 결과(S101), 새로운 트랜잭션이 시작하는 경우에, 해당 트랜잭션의 속성을 추출하여 리드(Read)인지를 판단한다(S102).

<83> 이때, 상기 판단 결과(S102), 상기 속성이 리드인 경우에, 상기 버스 스누프부(10)는 현재 트랜잭션 마스터 프로세서(7)(예컨대, 프로세서0)이외의 프로세서(예컨대, 프로세서1, 프로세서2 및 프로세서3)에서 공유가 요구(Shared Asserted)되는지를 판단한다(S103).

<84> 이때, 상기 판단 결과(S103), 상기 공유가 요구된 경우에, 해당 공유 요구에 어드레스 매치되는 프로세서가 존재(Match Asserted)하는지 판단하여(S104), 존재하지 않는 경우에, 상기 버스 스누프부(10)는 종료되고, 반면 존재하는 경우에, 상기 버스 스누프부(10)는 상기 트랜잭션 마스터 프로세서(7)의 해당 밸리드 비트를 인밸리드 상태(밸리드 비트 = 0)에서 밸리드 상태(밸리드 비트 = 1)로 설정(Set)하지 않으며 상기 어드레스에 대한 태그를 가지고 있던 매치된 프로세서에 대하여 해당 밸리드 비트를 밸리드 상태에서 인밸리드 상태로 클리어(Clear)한다(S105). 즉, 상기 캐시 메모리(6)에 존재하는 캐시 블록은 공유 상태이므로 시스템 이벤트가 발생하더라도 캐시 플러시 동작을 수행할 필요가 없다.

<85> 한편, 상기 판단 결과(S103), 상기 공유가 요구되지 않은 경우에, 즉 기존에 어떤 프로세서도 상기 어드레스에 대해서 캐시하지 않은 경우에, 밸리드 어레이부(30)는 디캔드(31)를 사용한 정치 동작을 통해 캐시 블록을 선택하여 캐시 블록 정보를 상기 버스 스누프부(10)로 제공한다.

<86> 이에, 상기 버스 스누프부(10)는 상기 정치 동작에 의한 캐시 블록 정보를 제공받아(S106), 선택된 특정 캐시 블록에서 상기 어드레스에 대응하는 인덱스에 의해 선정된 위치에 있는 태그 저장부(20)의 태그 램에 태그를 저장하며(S107), 상기 밸리드 어레이부(30)에서 해당 밸리드 비트를 인밸리드 상태에서 밸리드 상태로 설정한다(S108). 즉, 현재의 트랜잭션 마

스터 프로세서(7)가 해당 데이터를 배타적으로 보유하게 되므로 시스템 이벤트가 발생할 경우 캐시 플러시 동작을 수행할 필요가 생긴다.

<87> 이 때, 상술한 태그 저장 위치 선정은 상기 어드레스에 의해 해당되는 인덱스로 자동 결정되지만, 어떤 캐시 블록을 사용할 것인가의 선택은 상기 대캔드(31)를 사용하는데, 해당 디캔드(31)를 사용해 캐시 블록을 결정하는 것이 상술한 정치 동작(Placement Algorithm)이다.

<88> 이하, 도 13을 참조하여 상기 정치 동작을 설명한다.

<89> 도 13은 도 7의 디캔드를 이용한 정치 동작을 개념적으로 나타낸 도면이다.

<90> 캐시 블록0쪽이 하방향이고 캐시 블록7쪽이 상방향일 때 각 판단 단계에서 하방향이 선택될 때 '0'을 의미하고 상방향이 선택될 때 '1'을 의미한다고 하면, 상기 정치 동작은 대캔드(31)의 각 단계의 출력이 해당 브랜치(Branch)에 비어 있는 캐시 블록이 있음을 나타내는 '0'인 것을 조건으로 하여 하방향 캐시 블록을 찾아가는 것이다. 따라서, 도 13에 도시된 바와 같이, 맨 오른쪽으로부터 첫번째 AND 판단 단계에서 하방향이 '0'이므로 선택되어 해당 하방향 선택에 따라 '0'을 의미하고, 맨 오른쪽으로부터 두번째 AND 판단 단계에서 상방향이 '0'이므로 선택되어 해당 상방향 선택에 따라 '1'을 의미하며, 맨 오른쪽으로부터 세번째 AND 판단 단계에서 상방향이 '0'이므로 선택되어 해당 상방향 선택에 따라 '1'을 의미한다. 결국, '011'에 대응하는 캐시 블록3이 선택된다.

<91> 한편, 상기 판단 결과(S102), 상기 속성이 리드가 아닌 경우에, 상기 버스 스누프부(10)는 해당 속성이 변경 의도 리드(read-with-intent-to-modify)인지를 판단한다(S109).

<92> 이때, 상기 판단 결과(S109), 상기 속성이 변경 의도 리드인 경우에, 해당 변경 의도 리드에 어드레스 매치되는 프로세서가 존재하는지 판단하여(S110), 존재하는 경우에, 상기 버스

스누프부(10)는 기존에 상기 어드레스에 대한 태그를 가지고 있던 매치된 프로세서에 대하여 해당 밸리드 비트를 밸리드 상태에서 인밸리드 상태로 클리어 한다(S111). 즉 현재의 트랜잭션 마스터 프로세서(7)가 해당 데이터를 캐시하고 나서 변경할 의도를 가지고 있으므로 변경 상태가 될 것이고, 이에 따라 다른 프로세서는 상기 데이터에 대한 캐시 상태를 인밸리드 상태로 할 것이며, 상기 버스 스누프부(10)는 갱신 동작을 수행하여 해당 어드레스에 대하여 매치된 프로세서의 밸리드 비트를 클리어한다.

<93> 그리고, 상기 버스 스누프부(10)는 상기 트랜잭션 마스터 프로세서(7)에 대하여 상기 캐시 블록 정보 수신 단계(S106), 태그 저장 단계(S107) 및 밸리드 비트 설정 단계(S108)을 수행한다. 즉, 상기 정치 동작을 사용하여 상기 프로세서(7)에 대한 태그 저장부(20)의 태그 램과 밸리드 어레이부(30)의 밸리드 어레이를 갱신한다.

<94> 또한, 상기 판단 결과(S110), 상기 매치된 프로세서가 존재하지 않는 경우에도, 상기 버스 스누프부(10)는 상기 트랜잭션 마스터 프로세서(7)에 대하여 상기 캐시 블록 정보 수신 단계(S106), 태그 저장 단계(S107) 및 밸리드 비트 설정 단계(S108)를 수행한다.

<95> 한편, 상기 판단 결과(S109), 상기 속성이 변경 의도 리드가 아닌 경우에, 상기 버스 스누퍼부(10)는 상기 속성이 킬(Kill) 트랜잭션인지를 판단한다(S112).

<96> 이때, 상기 판단 결과(S112), 상기 킬 트랜잭션인 경우에, 상기 버스 스누프부(10)는 상기 변경 의도 리드인 경우와 동일한 동작을 수행하기 위해 상기 변경 의도 리드에 어드레스 매치되는 프로세서 존재 판단 단계로 진행하여(S110), 밸리드 비트 클리어 단계(S111), 캐시 블록 수신 단계(S106), 태그 저장 단계(S107) 및 밸리드 비트 설정 단계(S108)를 수행한다. 즉, 상기 킬 트랜잭션인 경우는 트랜잭션 마스터 프로세서가 될수 있는 현재의 버스 마스터 프로세서(7)가 공유 상태에 있던 캐시 데이터를 변경하고자 하는 것으로 해당 버스 마스터 프로세서

(7)의 캐시 메모리(6)에서 변경 상태로 천이할 것이다. 따라서, 상기 변경 의도 리드인 경우와 동일한 동작을 수행한다.

<97> 반면에, 상기 판단 결과(S112), 상기 킬 트랜잭션이 아닌 경우에, 상기 버스 스누프부(10)는 캐시 제어기의 교체(Replacement) 동작에 의해 캐스트 아웃(Cast Out)되어 스누프 푸시(Snoop Push)되었는지를 판단한다(S113).

<98> 이때, 상기 판단 결과(S113), 상기 캐스트 아웃되지 않아 스누프 푸시되지 않은 경우에, 상기 버스 스누프부(10)는 동작을 종료한다.

<99> 반면에, 상기 판단 결과(S113), 상기 캐스트 아웃되어 스누프 푸시된 경우에, 상기 버스 스누프부(10)는 기존에 상기 어드레스에 대한 태그를 가지고 있던 매치된 프로세서에 대하여 해당 밸리드 비트를 밸리드 상태에서 인밸리드 상태로 클리어 한다(S114).

<100> 즉, 상기 스누프 푸시는 세 가지 경우에 발생할 수 있는데, 하나는 다른 버스 마스터 프로세서(7)에 의한 리드에 스누프 히트(Snoop Hit)가 된 경우로서 이때는 캐시 메모리(6) 상태가 공유 상태로 천이될 것이다. 또 다른 하나는 다른 버스 마스터 프로세서(7)에 의한 라이트 또는 변경 의도 리드에 스누프 히트가 된 경우로서 이 때는 캐시 메모리(6) 상태가 인밸리드 상태로 천이할 것이다. 마지막으로 상기 캐시 제어기의 교체 동작에 의하여 캐스트 아웃되는 경우로서 이 또한 인밸리드 상태로 천이하게 된다. 따라서 상술한 스누프 푸시되는 세 가지 경우 모두 시스템 이벤트가 발생할 경우 해당 어드레스에 대해 캐시 플러시 동작을 수행할 필요가 없으므로, 해당 밸리드 비트를 클리어 한다.

<101> 상술한 상기 버스 스누프부(10)의 갱신 동작에 의하여 시스템 이벤트가 발생하였을 때 상기 캐시 플러시부(40)가 캐시 플러시 동작을 올바르게 수행할 수 있도록 상기 각 프로세서(7)의 캐시 메모리(6) 상태를 정확히 관리할 수 있게 된다.

<102> 이하, 도 14를 참조하여 상기 캐시 플러시 과정(S20)을 설명한다.

<103> 도 14는 도 11에 있어 캐시 플러시 과정을 나타낸 순서도이다.

<104> 상기 캐시 플러시부(40)에서 이벤트 검출기(42)는 사전에 정의된 캐시 플러시해야할 시스템 이벤트가 발생되었는지의 여부를 실시간으로 캐시 플러시 수행기(41)로 통보하고, 이에 해당 캐시 플러시 수행기(41)는 상기 이벤트 검출기(42)로부터 이벤트 발생 여부를 통보받는다(S201).

<105> 이에, 상기 캐시 플러시 수행기(41)는 상기 밸리드 어레이부(30)에서 상기 전체 프로세서(7)에대한 밸리드 어레이의 모든 밸리드 비트를 OR하여 전체 프로세서에 대한 밸리드 어레이 상태가 '1'로서 밸리드 상태(전체 밸리드 어레이 상태 = 1)인지를 확인한다(S202). 이때, 해당 밸리드 상태인 경우에, 상기 캐시 플러시 수행기(41)는 각 프로세서(7)의 캐시 메모리(6)에 대한 캐시 플러시 동작을 프로세서0부터 순서대로 진행하게 된다. 여기서, 상기 전체 프로세서에 대한 밸리드 어레이 상태란 해당 전체 프로세서에 대한 밸리드 어레이의 모든 밸리드 비트를 OR함으로써 알수 있다. 즉, 모든 밸리드 비트 중 하나만이라도 '1', 즉 밸리드 상태이면 해당 전체 프로세서에 대한 밸리드 어레이 상태는 '1', 즉 밸리드 상태가 된다.

<106> 구체적으로, 먼저, 상기 캐시 플러시 수행기(41)는 첫번째 프로세서N(N = 0)(S203)에 대한 밸리드 어레이 상태가 '1'로서 밸리드 상태(밸리드 어레이 상태(N) = 1)인지를 판단한다(S204). 이는 각 캐시 블록별로 수집된 디코아(32)의 최종 출력을 OR함으로써 얻을 수 있다.

<107> 이때, 상기 판단 결과(S204), 상기 프로세서의 밸리드 어레이 상태가 밸리드 상태가 아닌 경우, 즉 인밸리드 상태인 경우(밸리드 어레이 상태(N) = 0)에, 상기 캐시 플러시 수행기(41)는 다음 번째 프로세서N(N = N + 1)(S205)에 대한 상기 프로세서 밸리드 어레이 상태 판단 단계(S204)로 진행한다.

<108> 반면에, 상기 판단 결과(S204), 상기 밸리드 어레이 상태가 밸리드 상태인 경우(밸리드 어레이 상태(N) = 1)에, 상기 캐시 플러시 수행기(41)는 상기 프로세서의 첫번째 캐시 블록n(n = 0)(S206)에 대한 밸리드 어레이 상태가 '1'로서 밸리드 상태(밸리드 어레이 상태(N,n) = 1)인지를 판단한다(S207). 이는 각 캐시 블록의 디코아(32)의 최종 출력을 확인함으로써 판단할 수 있다.

<109> 이때, 상기 판단 결과(S207), 상기 캐시 블록의 밸리드 어레이 상태가 밸리드 상태가 아닌 경우, 즉 인밸리드 상태인 경우(밸리드 어레이 상태(N,n) = 0)에, 상기 캐시 플러시 수행기(41)는 다음 번째 캐시 블록n(n = n + 1)(S208)에 대하여 상기 캐시 블록 밸리드 어레이 상태 판단 단계(S207)로 진행한다.

<110> 반면에, 상기 판단 결과(S207), 상기 캐시 블록의 밸리드 어레이 상태가 밸리드 상태(밸리드 어레이 상태(N,n) = 1)인 경우에, 상기 캐시 플러시 수행기(41)는 어드레스 배열 동작(A^3 : Address Arrangement Algorithm)에 의해 추출된 값을 인덱스로 하고 해당 인덱스를 상기 태그 저장부(20)의 태그 램부(21)로 제공하여 해당 태그 저장부(20)의 매치 로직부(22)로부터 해당 인덱스에 대응하는 태그 정보를 제공받아 해당 인덱스와 태그를 매핑하여 결정된 해당 캐시 블록 어드레스를 추출한다(S209).

<111> 그리고 상기 캐시 플러시 수행기(41)는 상기 추출된 캐시 블록 어드레스에 대응하는 캐시 블록에 대한 리드 트랜잭션를 생성하여 캐시 버스 마스터(43)를 통해 프로세서 버스(12) 상

으로 출력해 해당 캐시 블록이 속해 있는 캐시 메모리(6)에 대한 프로세서(7)로 하여금 해당 캐시 메모리(6)의 캐시 블록에 대한 캐시 플러시를 수행하게 한다(S210).

<112> 여기서, 상기 어드레스 배열 동작은 디코아(32)를 사용하여 구현되는데, 이하 도 15를 참조하여 상기 어드레스 배열 동작을 설명한다.

<113> 도 15는 도 8의 디코아를 이용한 어드레스 배열 동작을 개념적으로 나타낸 도면.

<114> 도 15에서 '0b00000' 인덱스쪽이 하방향이고 '0b11111' 인덱스쪽이 상방향일 때, 각 판단 단계에서 하방향이 선택될 때 '0'을 의미하고 상방향이 선택될 때 '1'을 의미한다고 하면, 상기 어드레스 배열 동작은 디코아(32)의 각 단계의 출력이 해당 브랜치에 차 있는 캐시 블록이 있음을 나타내는 '1'인 것을 조건으로 하여 하방향 인덱스를 찾아가는 것이다. 따라서, 도 15에 도시된 바와 같이, 맨 오른쪽으로부터 첫번째 OR 판단 단계에서 하방향이 '1'이므로 선택되어 해당 하방향 선택에 따라 '0'을 의미하고, 맨 오른쪽으로부터 두번째 OR 판단 단계에서 하방향이 '1'이므로 선택되어 해당 하방향 선택에 따라 '0'을 의미하며, 맨 오른쪽으로부터 세번째, 네번째 및 다섯번째 OR 판단 단계에서 각각 상방향이 '1'이므로 선택되어 해당 상방향 선택에 따라 각각 '1'을 의미한다. 결국, '00111'을 의미하므로 '0b00111' 인덱스가 선택된다.

<115> 그런 후(S210), 상기 캐시 플러시 수행기(41)는 상기 캐시 플러시된 캐시 블록이 상기 리드 트랜잭션의 대상인 프로세서(7)에 대한 캐시 메모리(6)에 속해있는 캐시 블록 중 최종 캐시 블록인지를 판단한다(S211). 여기서, 최종 캐시 블록이란 상기 리드 트랜잭션의 대상인 프로세서(7)가 4개이고 각 프로세서(7)에 대한 캐시 모메리(6)에 속하는 캐시 블록이 8개라고할 때, 8번째 캐시 블록((M,N) = (M,7), 여기서 $0 \leq M \leq 3$)을 말한다.

<116> 이때, 상기 판단 결과(S211), 상기 최종 캐시 블록이 아닌 경우에, 상기 캐시 플러시 수행기(41)는 상기 프로세서(7)의 다음번째 캐시 블록(S208)에 대한 밸리드 어레이 상태 판단 단계(S207)로 진행한다.

<117> 반면에, 상기 판단 결과(S211), 상기 최종 캐시 블록인 경우에, 상기 캐시 플러시 수행기(41)는 상기 리드 트랜잭션의 대상인 프로세서(7)가 전체 프로세서 중 최종 프로세서인지를 판단한다(S212). 여기서, 최종 프로세서란 상기 예의 경우, 4번째 프로세서((M,N) = (3,N), 여기서 0 ≤N ≤7)를 말한다.

<118> 이때, 상기 판단 결과(S212), 상기 최종 프로세서가 아닌 경우에, 상기 캐시 플러시 수행기(41)는 다음번째 프로세서(S205)에 대한 밸리드 상태 판단 단계(S205)로 진행한다.

<119> 반면에, 상기 판단 결과(S212), 상기 최종 프로세서인 경우에, 상기 캐시 플러시 수행기(41)는 캐시 플러시 동작을 종료한다.

<120> 상술한 바와 같이, 상기 캐시 플러시 수행기(41)는 각 프로세서(7)의 모든 캐시 블록에서 캐시 플러시 동작을 완료하면 다시 처음부터 같은 동작을 반복 수행한다. 이는 각 프로세서(7)가 해당 캐시 메모리(6)를 디스에이블(Disable)시키기 전까지 계속되며, 해당 각 프로세서(7)에 대한 밸리드 비트는 각 프로세서(7)가 해당 캐시 메모리(6)를 디스에이블할 수 있는 근거로 삼게 된다.

<121> 또한, 본 발명에 따른 실시예는 상술한 것으로 한정되지 않고, 본 발명과 관련하여 통상의 지식을 가진 자에게 자명한 범위 내에서 여러 가지의 대안, 수정 및 변경하여 실시할 수 있다.

【발명의 효과】

<122> 이상과 같이, 본 발명은 최대 각 프로세서의 2차 캐시 메모리 크기만큼의 메모리 리드만을 수행함으로써, 프로세서 버스의 부하를 최소화할 수 있고, 캐시 플러시 자체가 특정 이벤트에 의하여 직접 트리거되는 캐시 플러시 알고리즘에 의하여 수행됨으로써, 캐시 플러시의 특정 이벤트에 대한 동시성을 보장할 수 있어 고속 자동화된 캐시 플러시가 이루어질 수 있게 하는 효과가 있다.

【특허청구범위】

【청구항 1】

각 프로세서의 캐시 메모리에서 배타 또는 변경의 밸리드 상태 캐시 블록에 대하여 갱신 동작 및 캐시 플러시 동작을 위한 캐시 블록 정보 및 인덱스 정보를 제공하는 밸리드 어레이부와;

태그 램을 저장하고 갱신 동작 및 캐시 플러시 동작을 위한 매치 어드레스 정보 및 태그 정보를 제공하는 태그 저장부와;

프로세서 버스를 감시하고 상기 각 캐시 메모리의 상태를 추적하여 상기 태그 저장부와 밸리드 어레이부에 대한 갱신 동작을 수행하는 버스 스누프부와;

시스템의 특정 이벤트를 감지하고 상기 캐시 메모리의 밸리드 상태 캐시 블록에 대해 캐시 플러시를 수행하게 하는 리드 트랜잭션을 생성하여 출력하는 캐시 플러시부를 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 시스템.

【청구항 2】

제 1 항에 있어서,

상기 태그 저장부는,

특정 인덱스에 대하여 일정한 수만큼의 태그를 저장하는 태그 램부와;

상기 태그 램부와 정합하여 상기 매치 어드레스 정보를 출력해 프로세서 버스 어드레스와 해당 어드레스 인덱스의 태그간 매치 여부를 제공하고 상기 태그 정보를 출력하여 리드 트랜잭션의 어드레스를 구성할 태그를 제공하는 매치 로직부를 포함하여 이루어진 것을 특징으로

하는 캐시 플러시 시스템.

【청구항 3】

제 1 항에 있어서,

상기 버스 스누프부는,

디캔드의 캐시 블록 정보를 이용한 정치 동작을 구동하여 상기 갱신 동작을 수행함을 특징으로 하는 캐시 플러시 시스템.

【청구항 4】

제 3 항에 있어서,

상기 정치 동작은,

상기 대캔드의 각 단계의 출력이 해당 브랜치에 비어 있는 캐시 블록이 있음을 나타내는 '0'인 것을 조건으로 하여 하방향 캐시 블록을 찾아감을 특징으로 하는 캐시 플러시 시스템.

【청구항 5】

제 1 항에 있어서,

상기 캐시 플러시부는,

시스템에 대한 특정 이벤트의 발생 여부를 검출하는 이벤트 검출기와;

상기 캐시 플러시 동작을 수행하여 상기 인덱스 정보와 상기 태그 정보로 매핑된 어드레스에 대응하는 캐시 블록에 대한 리드 트랜잭션을 생성하는 캐시 플러시 수행기와;

상기 생성된 리드 트랜잭션을 상기 프로세서 버스로 출력하여 상기 각 프로세서로 전달해 상기 캐시 메모리에 대한 캐시 플러시가 이루어지도록 하는 캐시 버스 마스터를 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 시스템.

【청구항 6】

제 5 항에 있어서,

상기 캐시 플러시 수행기는,

디코아의 인덱스 정보를 이용한 어드레스 배열 동작을 구동하여 상기 인덱스를 추출하고 해당 인덱스를 상기 태그 램부에 제공해 상기 매치 로직부로부터 태그 정보를 획득하고 해당 인덱스와 태그를 병합하여 상기 어드레스를 매핑함을 특징으로 하는 캐시 플러시 시스템.

【청구항 7】

제 6 항에 있어서,

상기 어드레스 배열 동작은,

상기 디코아의 각 단계의 출력이 해당 브랜치에 차 있는 캐시 블록이 있음을 나타내는 '1'인 것을 조건으로 하여 하방향 인덱스를 찾아감을 특징으로 하는 캐시 플러시 시스템.

【청구항 8】

프로세서 버스의 트랜잭션을 감시하고 각 프로세서 캐시 메모리의 상태를 추적하여 태그 램과 밸리드 어레이에 대한 갱신 동작을 수행하는 과정과;

특정 이벤트를 감지하고 상기 태그 램과 밸리드 어레이를 이용해 리드 트랜잭션을 생성 출력하여 상기 캐시 메모리 중 밸리드 상태의 캐시 블록에 대한 캐시 플러시 동작을 수행하는 과정을 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 방법.

【청구항 9】

제 8 항에 있어서,

상기 갱신 과정은,

상기 프로세서 버스 상에서 트랜잭션 시작을 감시하여 해당 트랜잭션의 속성을 추출해 리드인지를 판단하는 단계와;

상기 속성이 리드인 경우에 트랜잭션 마스터 프로세서이외의 프로세서에서 공유가 요구되는지를 판단하는 단계와;

상기 공유가 요구된 경우에 해당 공유 요구에 어드레스 매치되는 프로세서가 존재하는 경우 상기 트랜잭션 마스터 프로세서의 해당 밸리드 비트를 밸리드 상태로 설정하지 않으며 해당 매치된 프로세서의 해당 밸리드 비트를 인밸리드 상태로 클리어하는 단계를 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 방법.

【청구항 10】

제 9 항에 있어서,

상기 갱신 과정은,

상기 공유가 요구되지 않은 경우에 정치 동작에 의한 캐시 블록 정보를 제공받는 단계와;

상기 캐시 블록 정보에 따라 특정 캐시 블록에서 상기 어드레스에 대응하는 인덱스에 의해 선정된 위치에 태그를 저장하며 해당 캐시 블록의 밸리드 비트를 밸리드 상태로 설정하는 단계를 더 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 방법.

【청구항 11】

제 9 항에 있어서,

상기 갱신 과정은,

상기 속성이 리드가 아닌 경우에 해당 속성이 변경 의도 리드인지를 판단하는 단계와;

상기 속성이 변경 의도 리드인 경우와 변경 의도 리드가 아니면서 킬 트랜잭션인 경우에 해당 어드레스 매치되는 프로세서가 존재하는지 판단하여 존재하는 경우에 매치된 프로세서의 밸리드 비트를 인밸리드 상태로 클리어하고 존재하지 않는 경우에 클리어하지 않는 단계와;

정치 동작에 의한 캐시 블록 정보를 제공받는 단계와;

상기 캐시 블록 정보에 따라 특정 캐시 블록에서 상기 어드레스에 대응하는 인덱스에 의해 선정된 위치에 태그를 저장하며 해당 캐시 블록의 밸리드 비트를 밸리드 상태로 설정하는 단계를 더 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 방법.

【청구항 12】

제 11 항에 있어서,

상기 갱신 과정은,

상기 속성이 변경 의도 리드가 아니면서 킬 트랜잭션이 아닌 경우에 캐시 제어기의 교체 동작에 의해 캐스트 아웃되어 스누프 푸시되었는지를 판단하는 단계와;

상기 캐스트 아웃되어 스누프 푸시된 경우에 기존에 상기 어드레스에 대한 태그를 가지고 있던 매치된 프로세서에 대하여 해당 밸리드 비트를 인밸리드 상태로 클리어하는 단계를 더 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 방법.

【청구항 13】

제 8 항에 있어서,

상기 캐시 플러시 과정은,

이벤트 발생 여부를 통보받아 상기 전체 프로세서에 대한 밸리드 어레이 상태가 밸리드 상태인지를 확인하는 단계와;

첫번째 프로세서에 대한 밸리드 어레이 상태가 밸리드 상태인지를 판단하여 밸리드 상태인 경우에 상기 프로세서의 첫번째 캐시 블록에 대한 밸리드 어레이 상태가 밸리드 상태인지를 판단하는 단계와;

상기 캐시 블록의 밸리드 어레이 상태가 밸리드 상태인 경우에 어드레스 배열 동작에 의해 추출된 값을 인덱스로 하고 해당 인덱스에 대응하는 태그 정보를 제공받아 결정된 해당 캐시 블록 어드레스를 추출하는 단계와;

상기 추출된 캐시 블록 어드레스에 대응하는 캐시 블록에 대한 리드 트랜잭션를 생성하여 프로세서 버스 상으로 출력해 해당 캐시 블록록에 대한 캐시 플러시를 수행하게 하는 단계와;

상기 캐시 플러시된 캐시 블록이 상기 리드 트랜잭션의 대상 프로세서의 캐시 블록 중 최종 캐시 블록이면서 해당 프로세서가 전체 프로세서 중 최종 프로세서인 경우에 동작을 종료하는 단계를 포함하여 이루어진 것을 특징으로 하는 캐시 플러시 방법.

【청구항 14】

제 13 항에 있어서,

상기 캐시 플러시 과정은,

상기 프로세서의 밸리드 어레이 상태가 밸리드 상태가 아닌 경우와 상기 최종 프로세서가 아닌 경우에 다음 번째 프로세서에 대한 상기 프로세서 밸리드 어레이 상태 판단 단계로 진행함을 특징으로 하는 캐시 플러시 방법.

【청구항 15】

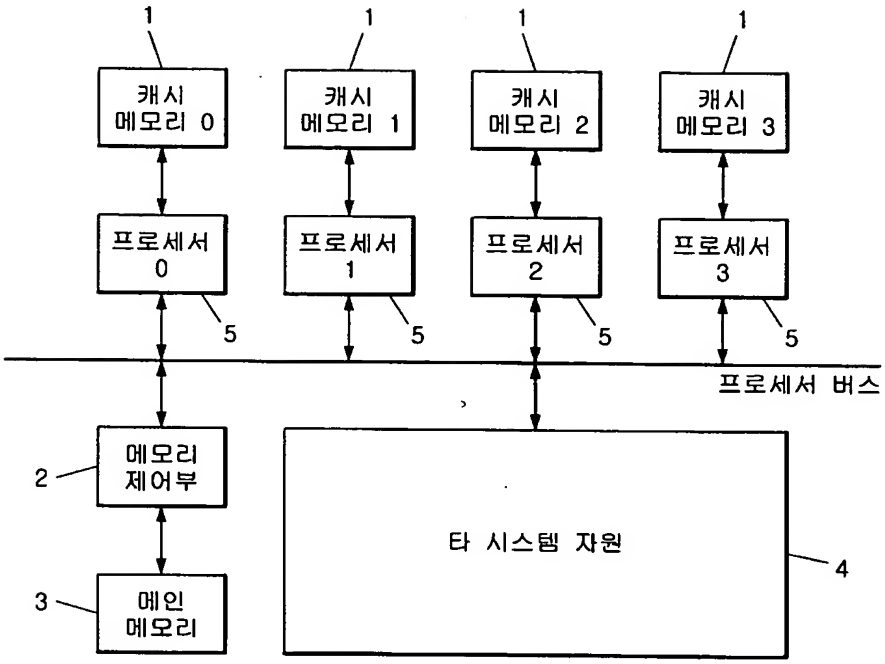
제 13항에 있어서,

상기 캐시 플러시 과정은,

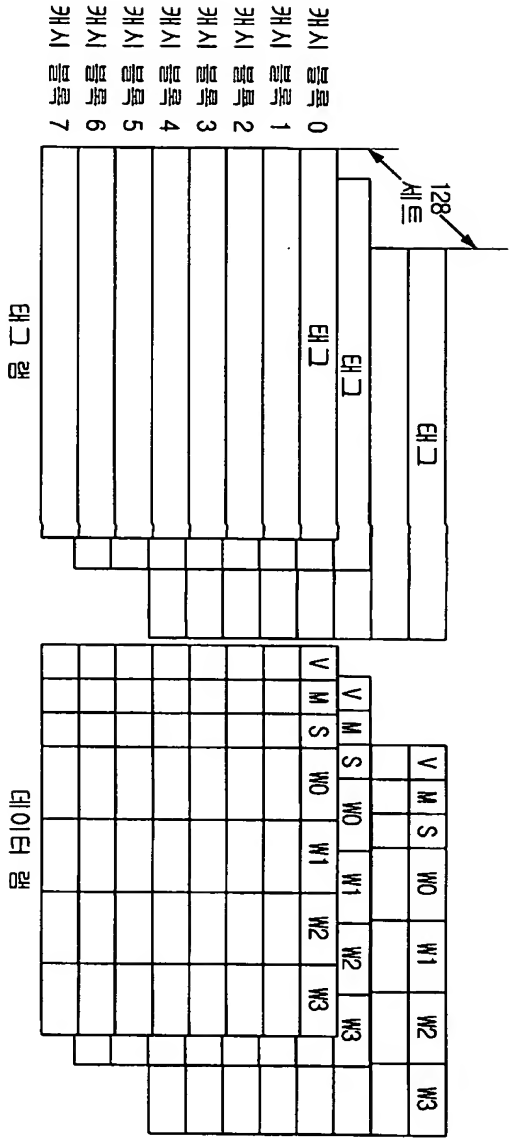
상기 캐시 블록의 밸리드 어레이 상태가 밸리드 상태가 아닌 경우와 상기 최종 캐시 블록이 아닌 경우에 다음 번째 캐시 블록에 대한 상기 캐시 블록 밸리드 어레이 상태 판단 단계로 진행함을 특징으로 하는 캐시 플러시 방법.

【도면】

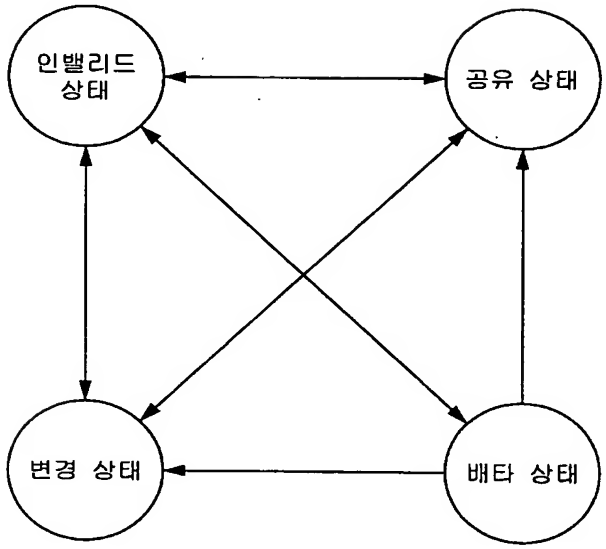
【도 1】


1
캐시 메모리 0
캐시 메모리 1
캐시 메모리 2
캐시 메모리 3
프로세서 0
프로세서 1
프로세서 2
프로세서 3
5
프로세서 버스
2
메모리 제어부
3
메인 메모리
타 시스템 자원
4

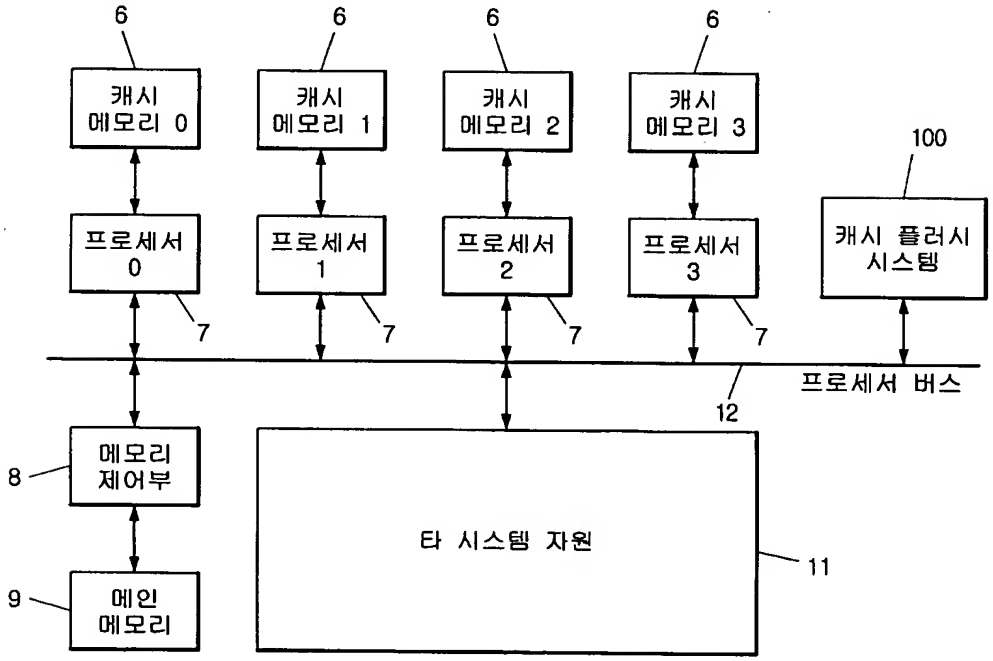
【도 2】


캐시 블록 0
캐시 블록 1
캐시 블록 2
캐시 블록 3
캐시 블록 4
캐시 블록 5
캐시 블록 6
캐시 블록 7
128 세트
태그
태그
태그 램
V
M
S
W0
W1
W2
W3
데이터 램

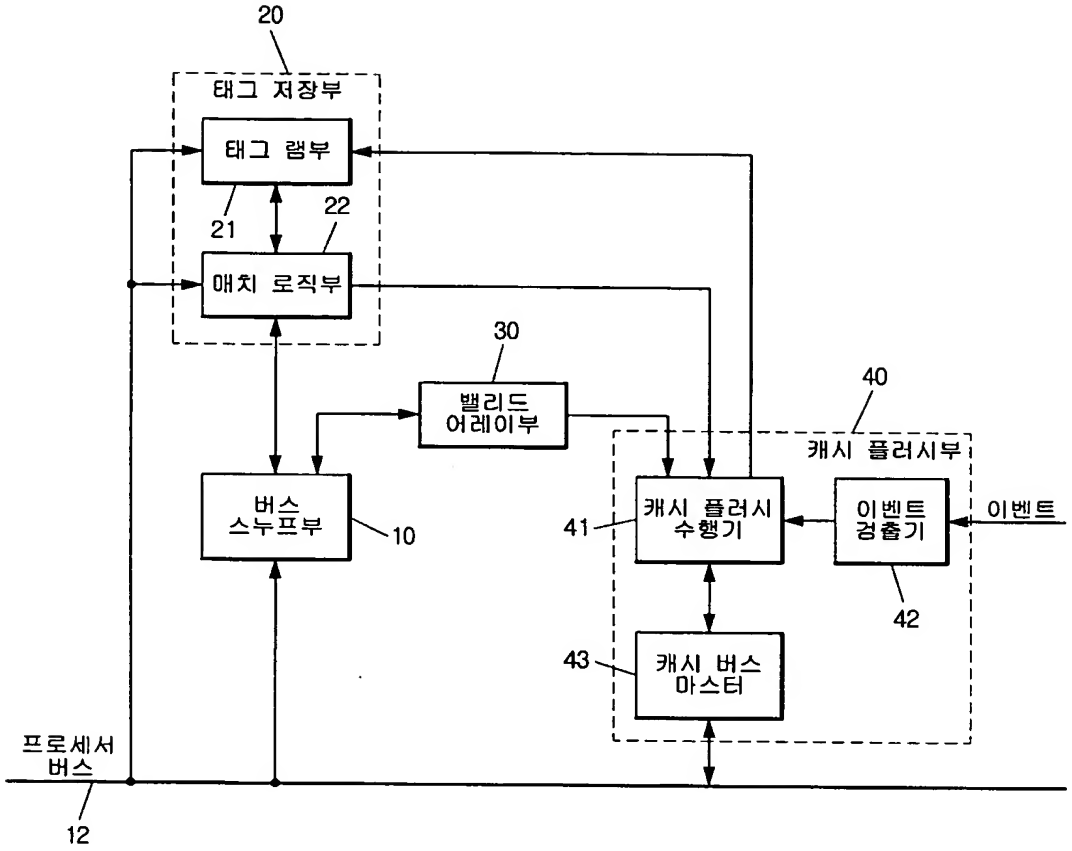
【도 3】


인밸리드 상태
공유 상태
변경 상태
배타 상태

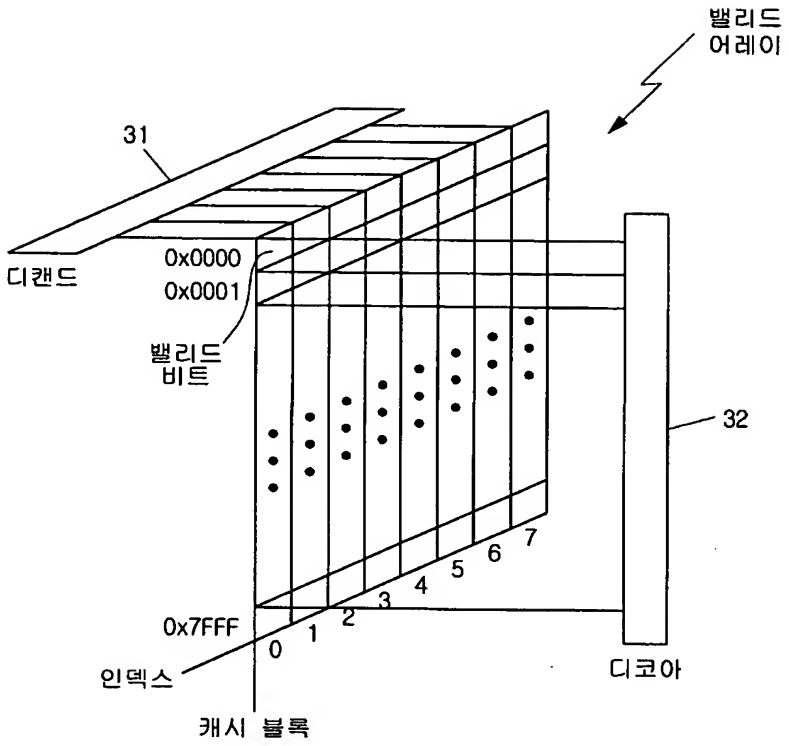
【도 4】


6
캐시 메모리 0
캐시 메모리 1
캐시 메모리 2
캐시 메모리 3
100
캐시 플러시 시스템
프로세서 0
프로세서 1
프로세서 2
프로세서 3
7
프로세서 버스
12
8
메모리 제어부
타 시스템 자원
11
9
메인 메모리

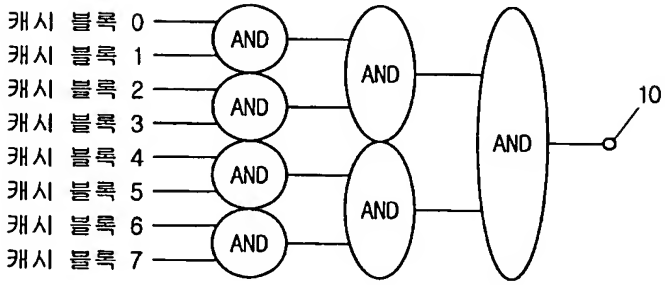
【도 5】


20
태그 저장부
태그 램부
21
22
애치 로직부
30
밸리드 어레이부
40
캐시 플러시부
버스 스누프부
10
41
캐시 플러시 수행기
이벤트 검출기
이벤트
42
43
캐시 버스 마스터
프로세서 버스
12

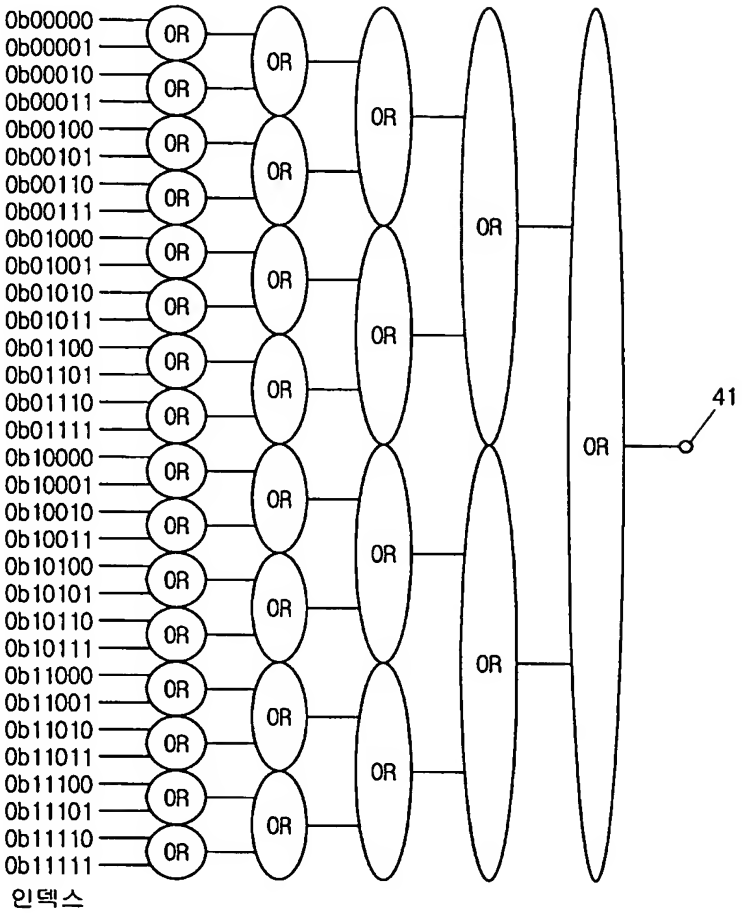
【도 6】


밸리드
어레이
31
디캔드
0x0000
0x0001
밸리드
비트
32
0
1
2
3
4
5
6
7
0x7FFF
인덱스
캐시 블록
디코아

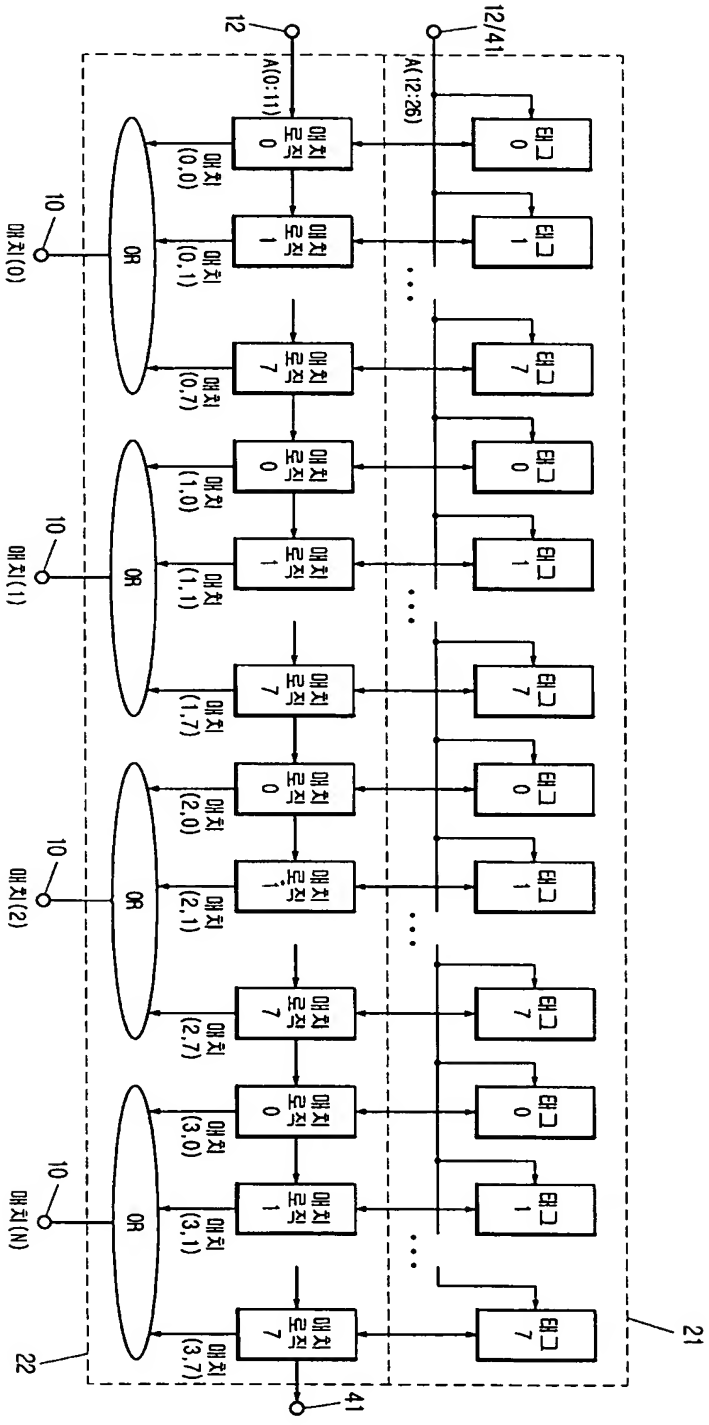
【도 7】


캐시 블록 0
캐시 블록 1
캐시 블록 2
캐시 블록 3
캐시 블록 4
캐시 블록 5
캐시 블록 6
캐시 블록 7
AND
AND
AND
AND
AND
AND
AND
10

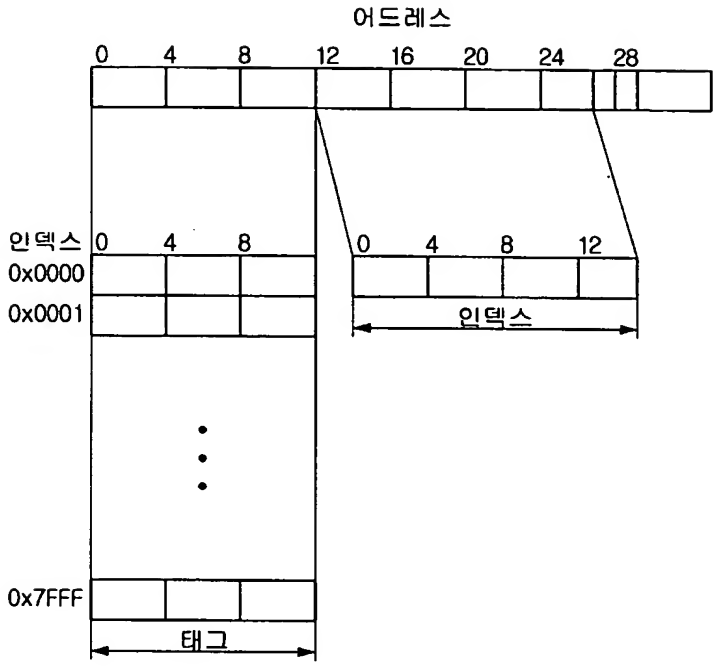
【도 8】


0b00000
0b00001
0b00010
0b00011
0b00100
0b00101
0b00110
0b00111
0b01000
0b01001
0b01010
0b01011
0b01100
0b01101
0b01110
0b01111
0b10000
0b10001
0b10010
0b10011
0b10100
0b10101
0b10110
0b10111
0b11000
0b11001
0b11010
0b11011
0b11100
0b11101
0b11110
0b11111
인덱스
OR
41

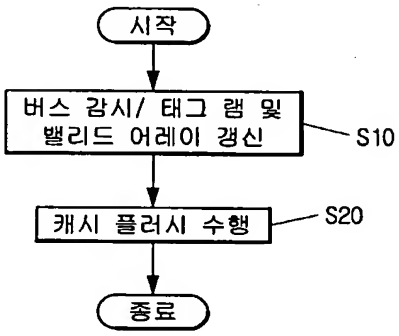
【도 9】


12
12/41
A(0:11)
A(12:26)
매치 로직 0
매치 로직 1
매치 로직 7
태그 0
태그 1
태그 7
OR
매치(0)
매치(1)
매치(2)
매치(N)
매치 (0,0)
매치 (0,1)
매치 (0,7)
매치 (1,0)
매치 (1,1)
매치 (1,7)
매치 (2,0)
매치 (2,1)
매치 (2,7)
매치 (3,0)
매치 (3,1)
매치 (3,7)
10
21
22
41

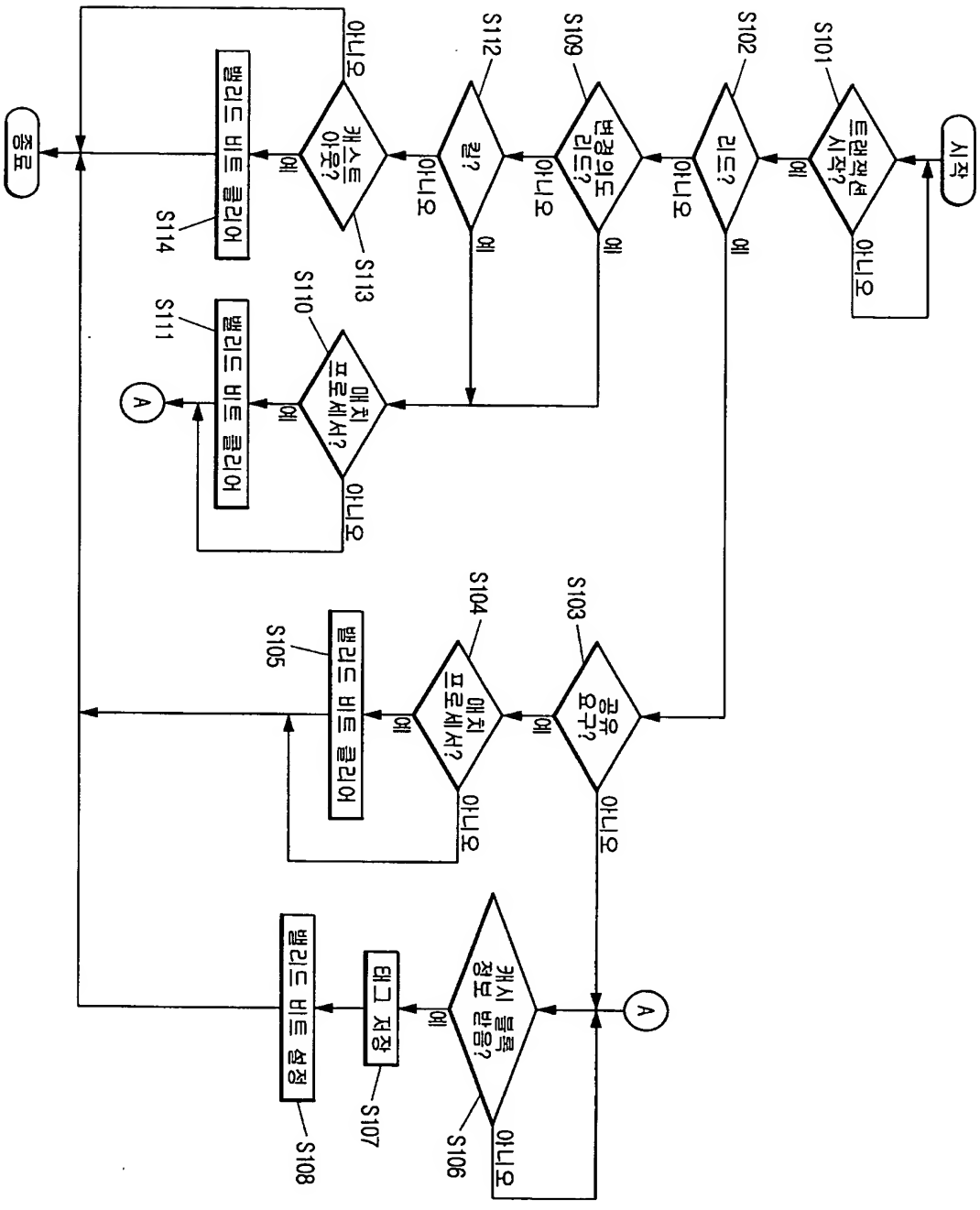
【도 10】


어드레스
0
4
8
12
16
20
24
28
인덱스
0
4
8
0
4
8
12
0x0000
0x0001
인덱스
0x7FFF
태그

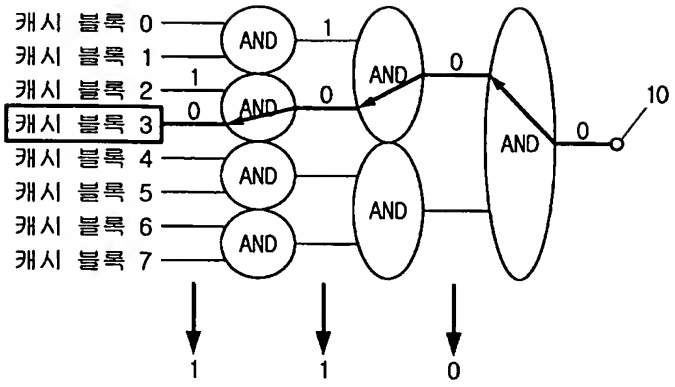
【도 11】


시작
버스 감시/ 태그 램 및 밸리드 어레이 갱신
S10
캐시 플러시 수행
S20
종료

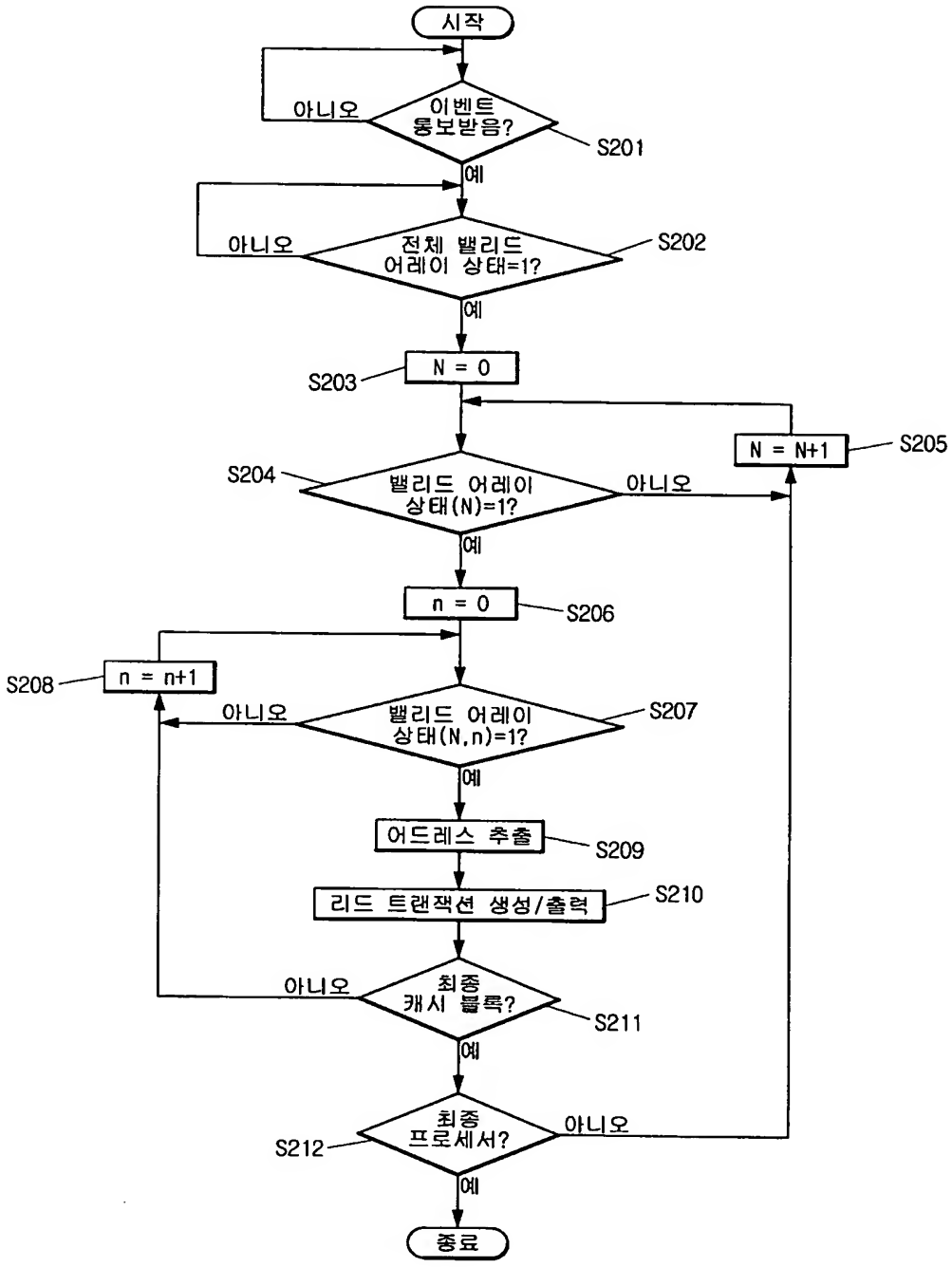
【도 12】


시작
트랜잭션 시작?
S101
예
아니오
리드?
S102
예
아니오
변경의도 리드?
S109
예
아니오
킬?
S112
예
아니오
캐스트 아웃?
S113
예
아니오
웰리드 비트 클리어
S114
종료
A
매치 프로세서?
S110
예
아니오
웰리드 비트 클리어
S111
A
공유?
S103
예
아니오
매치 프로세서?
S104
예
아니오
웰리드 비트 클리어
S105
A
캐시 블록 정보 발음?
S106
예
아니오
태그 저장
S107
웰리드 비트 설정
S108

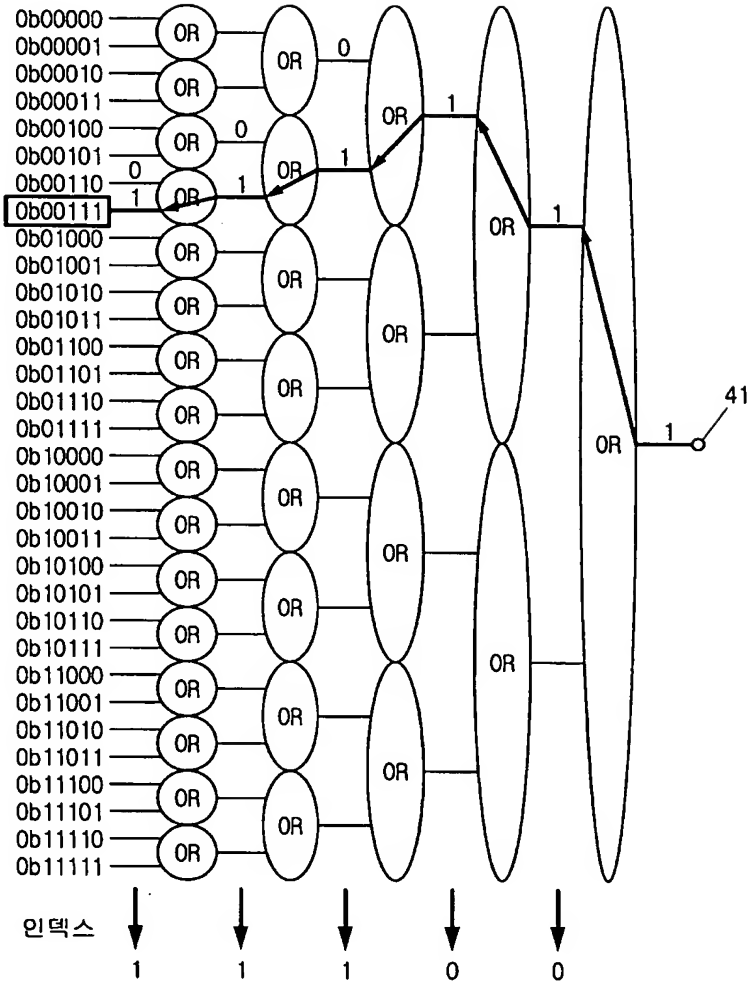
【도 13】


캐시 블록 0
캐시 블록 1
캐시 블록 2
캐시 블록 3
캐시 블록 4
캐시 블록 5
캐시 블록 6
캐시 블록 7
AND
AND
AND
AND
AND
AND
AND
1
1
0
0
0
0
0
10
1
1
0

【도 14】


시작
이벤트 통보받음?
아니오
S201
예
전체 밸리드 어레이 상태=1?
아니오
S202
예
N = 0
S203
N = N+1
S205
밸리드 어레이 상태(N)=1?
S204
아니오
예
n = 0
S206
n = n+1
S208
밸리드 어레이 상태(N,n)=1?
아니오
S207
예
어드레스 추출
S209
리드 트랜잭션 생성/출력
S210
최종 캐시 블록?
아니오
S211
예
최종 프로세서?
S212
아니오
예
종료

【도 15】


0b00000
0b00001
0b00010
0b00011
0b00100
0b00101
0b00110
0b00111
0b01000
0b01001
0b01010
0b01011
0b01100
0b01101
0b01110
0b01111
0b10000
0b10001
0b10010
0b10011
0b10100
0b10101
0b10110
0b10111
0b11000
0b11001
0b11010
0b11011
0b11100
0b11101
0b11110
0b11111
OR
0
1
41
인덱스
1
1
1
0
0